Lightweight Unsupervised Domain Adaptation by Convolutional Filter Reconstruction

Rahaf Aljundi, Tinne Tuytelaars

KU Leuven, ESAT-PSI - iMinds, Belgium

Abstract. Recently proposed domain adaptation methods retrain the network parameters and overcome the domain shift issue to a large extent. However, this requires access to all (labeled) source data, a large amount of (unlabeled) target data, and plenty of computational resources. In this work, we propose a lightweight alternative, that allows adapting to the target domain based on a limited number of target samples in a matter of minutes. To this end, we first analyze the output of each convolutional layer from a domain adaptation perspective. Surprisingly, we find that already at the very first layer, domain shift effects pop up. We then propose a new domain adaptation method, where first layer convolutional filters that are badly affected by the domain shift are reconstructed based on less affected ones.

1 Introduction

In recent years, great advances have been realized towards image understanding in general and object recognition in particular, thanks to end-to-end learning of convolutional neural networks, seeking the optimal representation for the task at hand. Unfortunately, performance remarkably decreases when taking the trained algorithms and systems out of the lab and into the real world of practical applications. This is known in the literature as the domain shift problem. The default solution is to retrain or finetune the system using additional training data, mimicking as close as possible the conditions during testing. However, such large labeled data is not always available.

Overcoming this domain shift problem without additional annotated data is the main goal of unsupervised domain adaptation. State-of-the-art methods for unsupervised domain adaptation of deep neural network architectures such as Domain Adversarial Training of Neural Networks(DANN) [1] and Learning Transferable Features with Deep Adaptation Networks (DAN) [2] proceed by adding new layers to the deep network or learning a joint architecture in order to come up with representations that are more general and informative across the source and target domains.

However, in spite of their good results on various benchmarks, these methods seem to be of limited value in a practical application. Indeed, deep adaptation methods require access to all the source data, a lot of computation time, a lot of resources, and a lot of unlabeled target data. This is in contrast to the typical

domain adaptation setting, where we want networks trained on big datasets such as Imagenet to be readily usable by different users and in a variety of settings.

So instead, we advocate the need for *light-weight domain adaptation schemes*, that require only a small number of target samples and can be applied quickly without heavy requirements on available resources, in an *on-the-fly* spirit. Using only a few samples, such a system could adapt to new conditions at regular time intervals, making sure the models are well adapted to the current conditions. The simpler sub-space based domain adaptation methods developed earlier for shallow architectures [3–5] seem good candidates for this setting. Unfortunately, when applied to the last fully connected layer of a standard convolutional neural network, they yield minimal improvement [6, 7]. However, the last layer might not be the best place to perform domain adaptation (DA). In this work, we start by analyzing the different layers of a deep network from a domain adaptation perspective. First, we show that *domain shift does not only affect the last layers of the network*, but can already manifest itself as early as the very first layer. Second, we show that the *filters exhibit different behavior in terms of domain shift*: while some filters result in a largely domain invariant representation, others lead to very different filter responce distributions for the source and target data. Based on this analysis, we propose a new light-weight domain adaptation method, focusing just on the first layer of the network.

2 Analysis of domain shift in the context of deep learning

Deep adaptation methods typically assume that the first layers are generic and need no adaptation, while the last layers are more specific to the dataset used for training and thus sensitive to the shift between the two domains. Therefore, most adaptation methods tend to adapt only the last layers and freeze the first layers [8, 2]. This assumption is based on the fact that the first layers mainly detect colors, edges, textures, etc. - features that are generic to all domains. On the other hand, in the last layers we can see high level information about the objects, which might be domain-specific. However, *even if the feature extraction method is generic, the features may still convey information about the domain.* This is indeed what one would expect for features that are not trained to be domain invariant. To understand how the domain shift evolves through the different layers, we perform an analysis of the output of each layer using Alexnet [9]. We use the standard adaptation benchmark, the Office dataset [10], specifically the two sets Amazon (A) and Webcam (W). This setup resembles the typical case of domain adaptation where the datasets are gathered from different resources. In addition, to study the low level shift, we created a gray scale set that contains the same images as in the Amazon dataset but converted to gray scale. We call it Amazon-Gray (AG). We consider two adaptation cases: A→AG and A→W. We start by fine-tuning AlexNet on the Amazon dataset (the source). Then, we consider each convolutional layer as an independent feature extraction step. Each layer is composed of a set of filter maps. We consider one filter from one layer at a time, and consider, for this analysis, the corresponding filter map as our

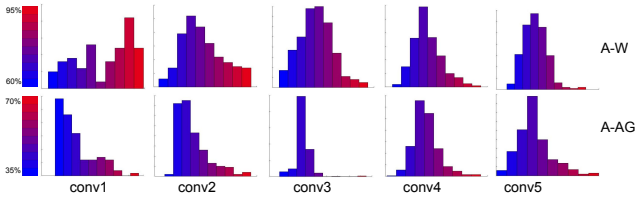

Fig. 1: The histograms of the filters' H-divergences for different layers.

feature space. Each instance (image) is a point in that space. For example, the first convolutional layer is composed of 96 filter maps, each of size [55x55]. We reshape each [55x55] filter map into a feature vector of length 3025 and consider this the feature representation of the image.

To quantify the domain shift, we study the H-divergence [11] w.r.t. each layer / filter and using linear SVM as our hypothesis. In Figure 1, we show the histograms of the filters' H-divergences w.r.t. each layer regarding the two study cases A→W and A→AG. We encode the value of the H-divergence by the color where blue indicates a low H-divergence (= "good" filters) while red indicates a high H-divergence (= "bad" filters).

Discussion From Figure 1, we can conclude that, in contrast to common belief, the *first layers are susceptible to domain shift even more than the later layers* (i.e., the distributions of the source and target filter outputs show bigger differences in feature space, resulting in larger H-divergence scores). Indeed, the filters of the first layers are similar to HOG, SURF or SIFT (edge detectors, color detectors, texture, etc.); they are generic w.r.t. different datasets, i.e. they give representative information regardless of the dataset. However, this information also conveys the specific characteristics of the dataset and thus the dataset bias. As a result, when the rest of the network processes this output, it will be affected by the shown bias, causing a degradation in performance.

Especially in the first layer of the convolutional neural network, we see large differences between different filters while in later layers the H-divergence of the filters follows a normal distribution as their input is affected by the domain shift in some of the first layer filters and thus it is harder to find non affected filters in later layers. Based on this analysis of the domain bias over different layers, we believe that *a good solution of the domain adaptation problem should start from the first layers* in order to correct each shift at the right level rather than waiting till the last layer and then trying to match the two feature spaces.

Our DA strategy Based on these findings we suggest: 1) to compute the divergence of the two datasets with respect to each filter as a measure for how good each filter is, and 2) since the filters of a convolutional layer are often correlated, to re-estimate the response map of the filters that are affected by the domain shift using those filters that are not. Here the goal is to obtain a new response map that resembles more the response of a source image for the affected filters. Instead of applying a threshold to decide on a set of good filters and bad filters and use the former set to re-estimate the later which might not

be correlated and thus not optimal, we decide which filters to reconstruct and which to use for their reconstruction in one step using a Lasso based optimization problem. Below, we first explain the divergence measure and then proceed to the core of the proposed method.

Divergence measure Instead of using the H-divergence that will add an extra cost due the need to train a domain classifier and also the need to have access to sufficient amount of data, we use the KL-divergence [12] [13] which is a measure of the difference between two probability distributions, in our case the source distribution P_S and the target distribution P_T. We estimate the probability distribution of the filter response given a small subset of the source data as input and likewise for the target data.

Filter selection As we explained before, we want to find the affected filters and the ones to be used in their reconstruction in one step. For that purpose we put each filter under a sanity check where we try to select the set of filters to be used in a regression function that predicts its output. Here, we do not consider the entire filter map, but rather the filter response at each point of the filter map separately, where, given the response of the other filters at this point we want to predict the current response. We use (a subset of) the source data as our training set. Going back to the literature, feature selection for regression has been studied widely. Lasso [14] and Elastic net [15] have shown good performance. We favor Lasso as it introduces the sparsity which is essential in our case to select as few and effective filters as possible. Having the response f_y and the set of predictors f_x, the main equation of Lasso can be written as follows:

$$B^* = argmin_B\{\sum_{i=1}^{n}(f_{y_i} - \beta_0 - \sum_{j=1}^{p} f_{j_{x_i}}\beta_j)^2 + \lambda\sum_{j=1}^{p}|\beta_j|\} \tag{1}$$

where β_0 is the residual, $B = \{\beta_j\}$ the estimated coefficients, n the number of source samples, p the number of filters, and λ a tuning parameter to control the amount of shrinkage needed. Clearly, if we have the response itself as a choice in the filters set, it will be directly selected as it is the most correlated with the output (i.e. itself). What we need to do next is to insert our additional selection criterion, i.e. the KL-divergence, where for each filter f_j, we have computed a KL divergence value, Δ_j^{KL}. We will use this divergence value to guide the selection procedure. This can be achieved by simply plugging the Δ_j^{KL} value in the L_1 norm regularization as follows:

$$B^* = argmin_B\{\sum_{i=1}^{n}(f_{y_i} - \beta_0 - \sum_{j=1}^{p} f_{j_{x_i}}\beta_j)^2 + \lambda\sum_{j=1}^{p}|\Delta_j^{KL}\cdot\beta_j|\} \tag{2}$$

Solving this optimization problem, we obtain the weights vector B^*, with a weight β_j^* for each filter f_j, including the filter we try to reconstruct. If the filter in hand has a non-zero weight, that means it is considered a good filter and we will keep its value. On the other hand, if the filter has zero weight then it will be marked for reconstruction and its set of filters with non-zero weights are used for this purpose.

Reconstruction After selecting the set of filters to be used for reconstruction $\{f_b\}$, we use the linear regression method to predict the filter output f_{b_y} given the responses of the selected filters. The linear regression is in its turn simple and efficient to compute. As a result, we obtain the final set of coefficients B_b for each bad filter f_b.
Prediction At test time we receive a target sample x_t. We pass it through the first layer and obtain the response of each filter map. Then, for each bad filter f_b, we use the responses of the selected set of filters to predict a source like response given the coefficients B_b. After that, we pass the reconstructed data to the next layer up to prediction.

3 Experiments

3.1 Setup & Datasets

Office benchmark [10]: we use three sets of samples: Webcam (W), DSLR (D) and Amazon (A). In addition, we use the gray scale version of the Amazon data (AG). The main task is object recognition. We use Alexnet [9] pretrained on Imagenet and fine-tuned on the Amazon data. We deal with three adaptation problems: A$\rightarrow$ AG, A $\rightarrow$ W and A $\rightarrow$ D. We do not consider D$\leftrightarrow$W, as with deep features the shift between the two sets is minimal.
Synthetic traffic signs$\rightarrow$ Dark illumination (Syn$\rightarrow$Dark): to imitate the real life condition, we train a traffic signs recognition network [16] on synthetic traffic signs [17] and test it on extreme dark cases that we extract from GSTRB [18]. In all experiments we used 10% of the target dataset as our available target samples and retain only 1% of the source for the adaptation purpose.

Baselines: We compare with the following baselines: **No adaptation(NA)** by testing the network fine-tuned on the source dataset directly on the target set without adaptation. **DDC** [8] adapts the last layer of AlexNet. **Subspace Alignment(SA)[3]** is a simple method, yet shows good performance. To make a fair comparison, we take the activations of the last fully connected layer before the classification layer and use them as features for this baseline. We perform the subspace alignment and retrain an SVM with a linear kernel on the aligned source data, then use the learned classifier to predict the labels of the target data. We also show the result of the SVM classifier trained on the source features before alignment(**SVM-fc**). We don't compare with deep DA methods such as DAN and DANN as they are not lightweight and don't fir our specifications.

3.2 Results and discussion

Table 1 shows the results achieved by different methods. In spite of the method's simplicity and the fact that it is just active on the first layer, we systematically improve over the raw performance obtained without domain adaptation. In the case of low level shift (Syn-Dark and in Amazon-Gray), the method adapts by anticipating the color information of the target dataset, i.e. reconstructing the

Table 1: Recognition accuracies of our method and the baselines

Method	A→W	A→D	A→AG	Syn→Dark
CNN(NA)	60.5	65.8	94.8	75.0
DDC[8]	61.8	64.4	-	-
SVM-fc(NA)	60.5	61.5	95.0	74.0
SA	61.8	61.5	95.2	76.1
Filter Reconstruction(Our)	**62.0**	**67.2**	**97.0**	**80.0**

color filters. In the case of Amazon-Webcam and Amazon-DSLR, the method tries to ignore the background of Webcam and DSLR datasets that is different from the white background in Amazon dataset. Of course in this case there is also a high level shift that can be corrected by adapting the last layer features. In Figure 2 we show a filter reconstruction example from Amazon-Gray target set where we have the original color image from Amazon and also each filter output which serves as a reference. The method also outperforms the DDC [8] which is dedicated to correct the shift at the last layer only as well as SA [3] where the method improvement was moderate.

Fig. 2: From left to right: a sample image from Amazon, the corresponding gray image, the output of a bad filter w.r.t. the color image, the output of the same bad filter w.r.t. the gray image and the reconstructed output.

4 Conclusion

In this work, we aim to push the limits of unsupervised domain adaptation methods to settings where we have few samples and limited resources to adapt, both in terms of memory and time. To this end, we perform an analysis of the output of a deep network from a domain adaptation point of view. We deduce that even though filters of the first layer seem relatively generic, domain shift issues already manifest themselves at this early stage. Therefore, we advocate that the adaptation process should start from the early layers rather than just adapting the last layer features, as is often done in the literature. Guided by this analysis, we propose a new method that corrects the low level shift without retraining the network. The proposed method is suitable when moving a system to a new environment and can be seen as a preprocessing step that requires just a few images to be functional.

Acknowledgment: This work was supported by the FWO project "Representations of and algorithms for the captation, visualization and manipulation of moving 3D objects, subjects and scenes". The first author PhD is funded by the FWO scholarship.

References

1. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. Journal of Machine Learning Research **17**(59) (2016) 1–35
2. Long, M., Wang, J.: Learning transferable features with deep adaptation networks. CoRR, abs/1502.02791 **1** (2015) 2
3. Fernando, B., Habrard, A., Sebban, M., Tuytelaars, T.: Unsupervised visual domain adaptation using subspace alignment. In: Proceedings of the IEEE International Conference on Computer Vision. (2013) 2960–2967
4. Gong, B., Grauman, K., Sha, F.: Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In: Proceedings of The 30th International Conference on Machine Learning. (2013) 222–230
5. Gopalan, R., Li, R., Chellappa, R.: Domain adaptation for object recognition: An unsupervised approach. In: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE (2011) 999–1006
6. Tommasi, T., Patricia, N., Caputo, B., Tuytelaars, T.: A deeper look at dataset bias. In: Pattern Recognition. Springer (2015) 504–516
7. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. arXiv preprint arXiv:1310.1531 (2013)
8. Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., Darrell, T.: Deep domain confusion: Maximizing for domain invariance. arXiv preprint arXiv:1412.3474 (2014)
9. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. (2012) 1097–1105
10. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: Computer Vision–ECCV 2010. Springer (2010) 213–226
11. Ben-David, S., Blitzer, J., Crammer, K., Pereira, F., et al.: Analysis of representations for domain adaptation. Advances in neural information processing systems **19** (2007) 137
12. Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Statist. **22**(1) (03 1951) 79–86
13. Kullback, S.: Letter to the editor: the kullback-leibler distance. (1987)
14. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological) (1996) 267–288
15. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **67**(2) (2005) 301–320
16. Cireşan, D., Meier, U., Masci, J., Schmidhuber, J.: Multi-column deep neural network for traffic sign classification. Neural Networks **32** (2012) 333–338
17. Moiseev, B., Konev, A., Chigorin, A., Konushin, A.: Evaluation of traffic sign recognition methods trained on synthetically generated data. In: Advanced Concepts for Intelligent Vision Systems, Springer (2013) 576–583
18. Sermanet, P., LeCun, Y.: Traffic sign recognition with multi-scale convolutional networks. In: Neural Networks (IJCNN), The 2011 International Joint Conference on, IEEE (2011) 2809–2813